

Browser-basierte Cache-Angriffe auf die RSA-Schlüsselgenerierung

Moritz Krebbel

Universität zu Lübeck
Institut für IT-Sicherheit

30. Oktober 2018

Motivation

Angriffe im Browser

Prime-and-Probe

Eviction-Set-Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung

Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick

- 1 Motivation
- 2 Angriffe im Browser
- 3 Portierung eines nativen Angriffs
- 4 Fazit und Ausblick

Motivation

Angriffe im Browser

Prime-and-Probe

Eviction-Set-Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung

Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick

- Seitenkanalangriffe auf die Mikroarchitektur sind mächtig
- Mehr potenzielle Ziele durch Angriff aus dem Browser
- Portierung nativer Angriffe zu browser-basierten
- Im Folgenden: Angriff auf den geteilten L3-Cache von Intel-CPUs
- Speicherzugriffe des Opferprogramms durch unterschiedliche Zugriffszeiten bei Cache-Hit und Miss

Algorithm 1: Pseudo-Code für Prime-and-Probe

```
1 Function primeAndProbe(evictionSet)  
2   foreach address in evictionSet do  
3     | readMem(address)  
4   wait()  
5   timestampBefore  $\leftarrow$  getTimestamp()  
6   foreach address in evictionSet do  
7     | readMem(address)  
8   | return getTimestamp() - timestampBefore > threshold
```

- Timer im Browser durch Counter-Thread

Eviction-Set-Suche

Motivation

Angriffe im Browser

Prime-and-Probe

Eviction-Set-Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung

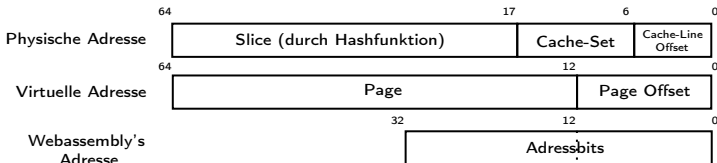
Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick



- Erzeuge Adresspool durch Allokation eines Buffers
- Wähle zufällige Adresse aus dem Pool als Zeugen
- Teilmenge des Pools ist Eviction-Set für den Zeugen
- Verbesserter Adresspool mittels colliding-addresses

Beschleunigung der Initialisierung

Motivation

Angriffe im Browser

Prime-and-Probe

Eviction-Set-Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung

Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick

- Nutzer besucht Website für wenige Minuten
- Reduzierung der Suchzeit (Median) durch optimierten Standardalgorithmus: 46 s \rightarrow 35 s
- Reduzierung der Suchzeit (Median) durch neuen Ansatz: 35 s \rightarrow 11 s
- Standardalgorithmus hat Ausreißer von über 100 Sekunden
- Durch Parallelisierung weitere Verbesserung möglich

Portierung eines Angriffs auf OpenSSL

Motivation

Angriffe im Browser

Prime-and-
Probe
Eviction-Set-
Suche
Ergebnisse

Portierung eines nativen Angriffs

**RSA-
Primzahl-
generierung**
Bremsen von
Code
Ein Thread
Mehrere
Threads
Bewertung

Fazit und Ausblick

- Nativer Angriff auf die Berechnung von $\gcd(e, p - 1)$ bzw. $\gcd(e, q - 1)$ in OpenSSL als Grundlage
- Primzahlrekonstruktion durch Überwachung von Shift- und Subtraktionsoperationen
- Leakage auch in Mozilla NSS ausnutzbar
- Problem: Ausführung einer Prime-and-Probe Operation bei Aktivität etwa 1500 Taktzyklen

Tabelle: Taktzyklendauer der Shift- und Subtraktionsoperation

Bitlänge	Shift	Subtraktion
2048	340	477
4096	577	718

Bremsen von Code im Browser

Motivation

Angriffe im Browser

Prime-and-Probe
Eviction-Set-Suche
Ergebnisse

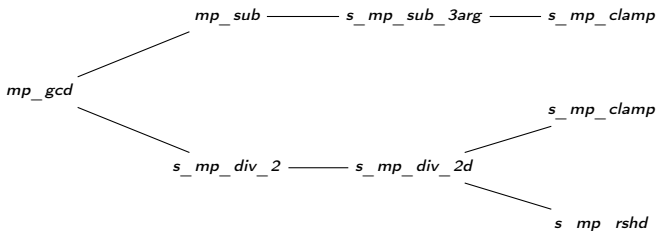
Portierung eines nativen Angriffs

RSA-Primzahl-generierung
Bremsen von Code

Ein Thread
Mehrere Threads
Bewertung

Fazit und Ausblick

- clflush-Instruktion im Browser nicht verfügbar
- Prime-and-Probe Operation ohne Zeitmessung ausführen
- Welcher Code-Teil soll gebremst werden?



Motivation

Angriffe im
BrowserPrime-and-
ProbeEviction-Set-
Suche

Ergebnisse

Portierung
eines nativen

Angriffs

RSA-
Primzahl-
generierungBremsen von
Code**Ein Thread**Mehrere
Threads

Bewertung

Fazit und
Ausblick

Funktion	Shift	Sub
Referenz	340	477
<i>*clamp₁</i>	541	608
<i>*rshd₂</i>	451	542
<i>mp_sub₇</i>	359	766
<i>*clamp_{1,2}</i>	542	557
<i>*clamp₁, *rshd₄</i>	643	549
<i>*clamp₁, *rshd₄, *div_2d₃</i>	623	560

- Unterschiedliche Cache-Lines bremsen unterschiedlich gut
- Besser mehrere Cache-Lines abwechselnd bremsen

Motivation

Angriffe im Browser

Prime-and-Probe

Eviction-Set-Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung

Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick

- Opferprogramm, Timer, Messung jeweils einen Thread
- Bremsung auf 4c/8t CPU mit nur einem physischen Kern

Funktion	Shift
Referenz	340
$*clamp_1 \& *clamp_1$	623
$*clamp_1 \& *rshd_4$	959
$*clamp_1 \& *rshd_4, *div_2d_3$	880
$*clamp_1, *rshd_4 \& *clamp_2, *div_2d_3$	745
$*clamp_1 \& *rshd_4 \& *div_2d_1$	1017
$*clamp_{1,2}, *rshd_4, *div_2d_{1,3,4}$ (6 Threads)	1109

- Besser nur eine Cache-Line pro Bremsthread

Motivation

Angriffe im
BrowserPrime-and-
ProbeEviction-Set-
Suche

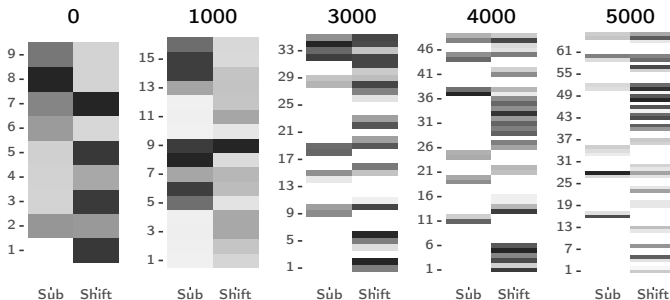
Ergebnisse

Portierung
eines nativen
AngriffsRSA-
Primzahl-
generierungBremsen von
Code

Ein Thread

Mehrere
Threads**Bewertung**Fazit und
Ausblick

- Überwachung der Folge 4s2s1s5s2s1 in einem Thread



- Bremsung von mindestens 3000 Taktzyklen oder mehr Kerne zur Unterscheidung nötig

Motivation

Angriffe im Browser

Prime-and-Probe

Eviction-Set-Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung

Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick

- Angriffs-Initialisierung unter 10 Sekunden möglich
- Limitierte Möglichkeiten erschweren Portierung von Angriffen im Browser
- Wiedereinführung hoch-präziser Timer ermöglicht Angriffe
- Standard-Webtechnologien ermöglichen Angriffe auf viele Endgeräte
- Cache-Angriffe werden noch Jahre Bestand haben

Weitere Bremsansätze

Motivation

Angriffe im Browser

Prime-and-Probe
Eviction-Set-Suche
Ergebnisse

Portierung eines nativen Angriffs

RSA-Primzahl-generierung
Bremsen von Code
Ein Thread
Mehrere Threads
Bewertung

Fazit und Ausblick

- Implementierung von Prime-and-Probe in Javascript bringt 10% Performancegewinn
- Standard für Prime-and-Probe ist Pointer-Chasing
- Aufteilung in zwei Ketten erlaubt 50% mehr Messungen pro Zeitintervall
- Pointer-Array statt Pointer-Chasing bremst besser wenn viele Cache-Lines in einem Thread gebremst werden (1 Thread bremst Shift auf ca. 800 Taktzyklen)
- Aufteilen des Pointer-Arrays ist nachteilig
- Weglassen von Back-Probing bringt nur Vorteile bei Pointer-Array

Beschleunigung der Eviction-Set-Suche

Motivation

Angriffe im Browser

Prime-and- Probe

Eviction-Set- Suche

Ergebnisse

Portierung eines nativen Angriffs

RSA- Primzahl- generierung

Bremsen von Code

Ein Thread

Mehrere Threads

Bewertung

Fazit und Ausblick

- Verfeinerung des Adresspools mittels Leakage

Algorithm 2: Pseudo-Code für colliding-address-Suche

```
1 Function FindCollidingAddresses()  
2     buffer = malloc(Size)  
3     for p from 60 to Size-1 do  
4         for i from 60 to 0 do  
5             | buffer[p-i]  $\leftarrow$  0  
6             timestampBefore  $\leftarrow$  getTimestamp()  
7             read x  
8             result[p]  $\leftarrow$  getTimestamp() - timestampBefore
```
